We claim:

1. A method of processing an application request on an end user application and an application server comprising the steps of:

a) initiating the application request on the end user application in a first language with a first application program;

b) transmitting the application request to the server and converting the application request from the first language of the first end user application to a language running on the application server;

c) processing said application request on the application server;

d) transmitting a response to the application request from the application server to the end user application, and converting theresponse to the application request from the language running on the application server to the first language of the first end user application; and

e) wherein the end user application and the application server have at least one connector therebetween, and the steps of (i) converting the application request from the first language of the first end user application as a source language to the language running on the application server as a target language, and (ii) converting a response to the application request from the language running on the application server as a source language to the first language of the first end user application as a target language, each comprise the steps of:

1) invoking connector metamodels of respective source and target languages;

2) populating the connector metamodels with metamodel data of each of the respective source and target languages; and

3) converting the source language to the target language.

2. The method of claim 1 wherein the end user application is a web browser.

3. The method of claim 2 wherein the end user application is connected to the application server through a web server, and the web server comprises an connector.

4. The method of claim 1 wherein the metamodel metadata comprises invocation metamodel metadata, application domain interface metamodel metadata, and type descriptor metamodel metadata.

5. The method of claim 4 wherein the invocation metamodel metadata is chosen from the group consisting of message control information, security data, transactional semantics, trace and debug information, pre-condition and post-condition resources, and user data.

6. The method of claim 4 wherein the application domain interface metamodel metadata comprises input parameter signatures, output parameter signatures, and return types.

7. The method of claim 4 wherein the application domain interface metamodel metadata further includes language metamodel metadata.

8. The method of claim 7 wherein the language metamodel metadata includes mappings between source and target languages.

9. The method of claim 8 wherein one of the source or target languages is object oriented, the other of the target or source languages is not object oriented, and the language metamodel metadata maps encapsulated objects into code and data.

10. The method of claim 9 wherein the language metamodel metadata maps object inheritances into references and pointers

11. The method of claim 8 wherein the source language and the target language are different object oriented languages, and the language metamodel metadata maps encapsulated code and data between the languages.

12. The method of claim 4 wherein the type descriptor metamodel metadata defines physical realizations, storage mapping, data types, data structures, and realization constraints.

13. The method of claim 1 wherein the transaction is a rich transaction comprising a plurality of individual transactions, and further comprising processing the plurality of individual transactions on one end user application and a plurality of application servers.

14. The method of claim 13 comprising passing individual transactions among individual application servers.

15. The method of claim 1 wherein one of the sourcelanguage or the target language is chosen from the group consisting of C and C++.

16. A transaction processing system comprising a client, a server, and at least one connector therebetween,

a) the client having an end user application, and being controlled and configured to initiate an application request with the server in a first language with a first application program and to transmit the application request to the server;

b) the connector being configured and controlled to receive the application request from the client, convert the application request from the first language of the first end user application running on the client to a language running on the server;

c) the server being configured and controlled to receive the converted application request from the connector and processing the said application request in a second language with a second application program residing on the server, and to thereafter transmit a response to the application request through the connector back to the first application program on the client;

d) the connector being configured and controlled to receive a response to the application request from the server, to convert a response to the application request from the language running on the application server to the first language of the first application program running on the client; and

e) wherein connector between the client and the server is configured and controlled to (i) convert the application request from the first language of the client application on the client as a source language to the language running on the application server as a target language, and (ii) convert the response to the application request from the language running on the application server as a source language to the first language of the client application running on the client as a target language, each by a method comprising the steps of:

1) retrieving connector metamodels of respective source and target languages from a metamodel metadata repository;

2) populating the connector metamodels with metamodel data from the metamodel metadata repository for each of the respective source and target languages; and

3) invoking the retrieved, populated connector metamodels and converting the source language to the target language.

17. The system of claim 16 wherein the end user application is a web browser.

18. The system of claim 17 wherein the end user application is connected to the application server through a web server, and the web server comprises an connector.

19. The system of claim 16 wherein the metamodel metadata comprises invocation metamodel metadata, application domain interface metamodel metadata, and type descriptor metamodel metadata.

20. The system of claim 19 wherein the invocation metamodel metadata is chosen from the group consisting of message control information, security data, transactional semantics, trace and debug information, pre-condition and post-condition resources, and user data.

21. The system of claim 19 wherein the application domain interface metamodel metadata comprises input parameter signatures, output parameter signatures, and return types.

22. The system of claim 19 wherein the application domain interface metamodel metadata further includes language metamodel metadata.

23. The system of claim 22 wherein the language metamodel metadata includes mappings between source and target languages.

24. The system of claim 23 wherein one of the source or target languages is object oriented, the other of the target or source languages is not object oriented, and the language metamodel metadata maps encapsulated objects into code and data.

25. The system of claim 24 wherein the language metamodel metadata maps object inheritances into references and pointers

26. The system of claim 24 wherein the source language and the target language are different object oriented languages, and the language metamodel metadata maps encapsulated code and data between the languages.

27. The system of claim 20 wherein the type descriptor metamodel metadata defines physical realizations, storage mapping, data types, data structures, and realization constraints.

28. The system of claim 16 wherein said system has a plurality of application servers and is configured and controlled to process rich transactions.

29. The system of claim 16 wherein one of the source or tagret languages is chosen from the group consising of C and C++.

30. A transaction processing system configured and controlled to interact with a client application, and comprising a server, and at least one connector between the server and the client application, where the client has an end user application, and is controlled and configured to initiate an application request with the server in a first language with a first application program and to transmit theapplication request to the server, wherein:

a) the connector being configured and controlled to receive an application request from the client, convert the application request from the first language of the first end user application running on the client to a language running on the server;

b) the server being configured and controlled to receive the converted application request from the connector and process the said application request in a second language with a second application program residing on the server, and to

thereafter transmit a response to the application request through the connector back to the first application program on the client;

c) the connector being configured and controlled to receive the application request from the server, to convert a response to the application request from the language running on the application server to the first language of the first application program running on the client; and

d) wherein connector between the client and the server is configured and controlled to (i) convert the application request from the first language of the client application on the client as a source language to the language running on the application server as a target language, and (ii) convert the response to the application request from the language running on the application server as a source language to the first language of the client application running on the client as a target language, each by a method comprising the steps of:

1) retrieving connector metamodel metadata of respective source and target languages from a metamodel metadata repository;

2) populating the connector metamodels with metamodel data of each of the respective source and target languages from the metamodel metadata repository and invoking the retrieved, populated connector metamodels; and

3) converting the source language to the target language.

31. The system of claim 30 wherein the end user application is a web browser.

32. The system of claim 31 wherein the end user application is connected to the application server through a web server, and the web server comprises an connector.

33. The system of claim 30 wherein the metamodel metadata comprises invocation metamodel metadata, application domain interface metamodel metadata, and type descriptor metamodel metadata.

34. The system of claim 33 wherein the invocation metamodel metadata is chosen from the group consisting of message control information, security data, transactional semantics, trace and debug information, pre-condition and post-condition resources, and user data.

35. The system of claim 33 wherein the application domain interface metamodel metadata comprises input parameter signatures, output parameter signatures, and return types.

36. The system of claim 33 wherein the application domain interface metamodel metadata further includes language metamodel metadata.

37. The system of claim 36 wherein the language metamodel metadata includes mappings between source and target languages.

38. The system of claim 37 wherein one of the source or target languages is object oriented, the other of the target or source languages is not object oriented, and the language metamodel metadata maps encapsulated objects into code and data.

39. The method of claim 37 wherein the source language and the target language are different object oriented languages, and the language metamodel metadata maps encapsulated code and data between the languages.

40. The system of claim 39 wherein the language metamodel metadata maps object inheritances into references and pointers

41. The system of claim 33 wherein the type descriptor metamodel metadata defines physical realizations, storage mapping, data types, data structures, and realization constraints.

42. The system of claim 30 wherein said system has a plurality of application servers and is configured and controlled to process rich transactions.

43. The system of claim 30 wherein one of the source or target languages is chosen from the group consisting of C and C++?

44. A groupware system having a plurality of end user applications, said end user applications each comprising an e-mail client, a content database client, and a content replication client, said system further comprising an e-mail server, a content database server, and a content replication server, said groupware system being configured and controlled to communicate between disparate end user applications, said groupware system comprising at least one connector between a server and an end user application, wherein the end user application is controlled and configured to participate with a server in a first language with a first application program and the server is configured and controlled to participate with the client in a second language, and wherein:

a) the connector is configured and controlled to receive an application request from the end user application, convert the application request from the first language of the first end user application to a language running on the server;

b) the server being configured and controlled to receive the converted application request from the connector and process the said application request in a second language with a second application program residing on the server, and to thereafter transmit the response to the application request through the connector back to the end user application;

c) the connector being configured and controlled to receive the response to the application request from the server, to convert a response to the application request from the language running on the application server to the first language of the end user application; and

d) wherein connector between the end user application program and the server is configured and controlled to (i) convert the application request from the first language of the end user application as a source language to the language running on the application server as a target language, and (ii) convert the response to the application request from the language running on the application server as a source language to the language of the end user application as a target language, each by a method comprising the steps of:

1) retrieving connector metamodel metadata of respective source and target languages from a metamodel metadata repository;

2) populating the connector metamodels with metamodel data of each of the respective source and target languages from the metamodel metadata repository and invoking the retrieved, populated connector metamodels; and

3) converting the source language to the target language.

45. The groupware system of claim 43 wherein the metamodel metadata comprises invocation metamodel metadata, application domain interface metamodel metadata, and type descriptor metamodel metadata.

46. The groupware system of claim 43 wherein the invocation metamodel metadata is chosen from the group consisting of message control information, security data, transactional semantics, trace and debug information, pre-condition and post-condition resources, and user data.

47. The groupware system of claim 45 wherein the application domain interface metamodel metadata comprises input parameter signatures, output parameter signatures, and return types.

48. The groupware system of claim 45 wherein the application domain interface metamodel metadata further includes language metamodel metadata.

49. The groupware system of claim 48 wherein the language metamodel metadata includes mappings between source and target languages.

50. The groupware system of claim 49 wherein one of the source or target languages is object oriented, the other of the target or source languages is not object oriented, and the language metamodel metadata maps encapsulated objects into code and data.

51. The groupware system of claim 49 wherein the source language and the target language are different object oriented languages, and the language metamodel metadata maps encapsulated code and data between the languages.

52. The system of claim 49 wherein the language metamodel metadata maps object inheritances into references and pointers

53. The system of claim 44 wherein the type descriptor metamodel metadata defines physical realizations, storage mapping, data types, data structures, and realization constraints.

54. The system of claim 44 wherein one of the source or target languages is chosen from the group consisting of C and C++.

55. A program product comprising a storage medium having invocation metamodel metadata, application domain interface metamodel metadata, and language metamodel

metadata, and computer instructions for building a metamodel metadata repository of source and target language metamodel metadata.

56. The program product of claim 55 further comprising computer instructions for building connector stubs from said metamodel metadata.

57. The program product of claim 55 further comprising computer instructions to build a connector for carrying out the steps of:

1) retrieving connector metamodel metadata of respective source and target languages from the metamodel metadata repository;

2) populating the connector metamodels with metamodel data of each of the respective source and target languages from the metamodel metadata repository and invoking the retrieved, populated connector metamodels; and

3) converting the source language to the target language.

58. The program product of claim 57 wherein the metamodel metadata in the repository comprises invocation metamodel metadata, application domain interface metamodel metadata, and type descriptor metamodel metadata.

59. The program product of claim 58 wherein the invocation metamodel metadata is chosen from the group consisting of message control information, security data, transactional semantics, trace and debug information, pre-condition and post-condition resources, and user data.

60. The program product of claim 58 wherein the application domain interface metamodel metadata comprises input parameter signatures, output parameter signatures, and return types.

61. The program product of claim 58 wherein the application domain interface metamodel metadata further includes language metamodel metadata.

62. The program product of claim 61 wherein the language metamodel metadata includes mappings between source and target languages.

63. The program product of claim 62 wherein one of the source or target languages is object oriented, the other of the target or source languages is not object oriented, and the language metamodel metadata maps encapsulated objects into code and data.

64. The program product of claim 62 wherein the source language and the target language are different object oriented languages, and the language metamodel metadata maps encapsulated code and data between the languages.

65. The program product of claim 64 wherein the language metamodel metadata maps object inheritances into references and pointers

66. The program product of claim 58 wherein the type descriptor metamodel metadata defines physical realizations, storage mapping, data types, data structures, and realization constraints.

67. The program product of claim 58 wherein one of the source and target languages is chosen from the group consisting of C and C++.